PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 12/08 // 11/00

A2

(11) International Publication Number: WO 98/13763

(43) International Publication Date: 2 April 1998 (02.04.98)

(21) International Application Number: PCT/IB97/01146

(22) International Filing Date: 23 September 1997 (23.09.97)

(30) Priority Data:
08/719,609 25 September 1996 (25.09.96) US

(71) Applicant: PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(71) Applicant *(for SE only)*: PHILIPS NORDEN AB [SE/SE]; Kottbygatan 7, Kista, S-164 85 Stockholm (SE).

(72) Inventor: JACOBS, Eino; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(74) Agent: DE HAAS, Laurens, J.; Internationaal Octrooibureau B.V., P.O. Box 220, NL-5600 AE Eindhoven (NL).

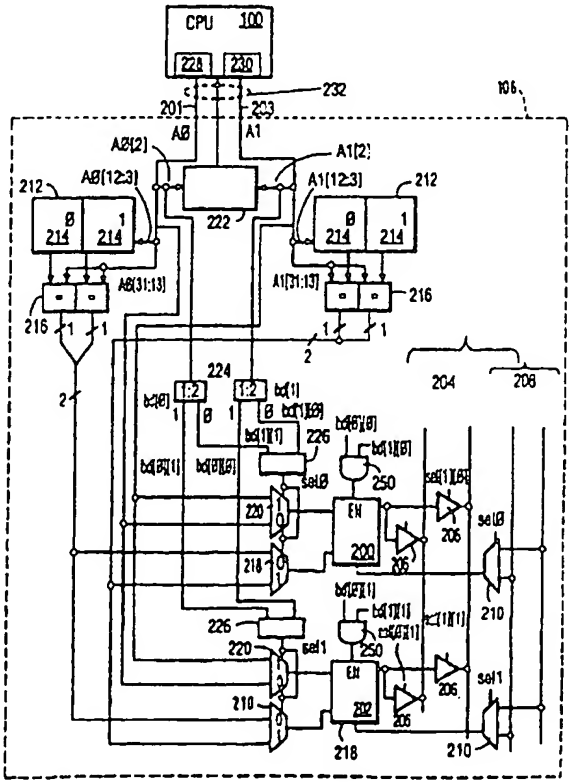
(81) Designated States: JP, KR, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: MULTI-PORT CACHE MEMORY WITH ADDRESS CONFLICT DETECTION

(57) Abstract

A multi-port cache memory is disclosed. The multi-port cache operates in a microprocessor system, and includes multiple memory banks and multiple ports for enabling accesses to the banks. Conflict detection circuitry detects simultaneous addressing of a first memory bank through a first port and a second port, and stalls microprocessor operations for a predetermined number of clock cycles in response to the detection of simultaneous addressing. Conflict resolution circuitry allows access to the first bank through the first port during the stall, and allows access through the second port after the stall is complete. Generally, the conflict resolution circuitry allows access through ports that are attempting to access the first memory bank in order of ascending priority during successive clock cycles while the microprocessor is stalled. One or more of the ports attempting to access the first bank may be allowed access before or after the time the microprocessor is stalled. Each bank is single-ported. The banks have non overlapping address spaces, and are addressed so that words within a cache block are distributed among multiple banks.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

MULTI-PORT CACHE MEMORY WITH ADDRESS CONFLICT DETECTION

The present invention relates to a processing system with a cache memory, and more particularly to a cache having multiple access ports.

A cache is a small, fast memory placed between a processor and main memory in order to reduce the effective time required by a processor to access addresses, instructions or data that are normally stored in main memory. For example, when a processor reads a word from main memory, the word and neighbouring words are read as a block from main memory into the cache. Typically, there is a high probability that the processor will next attempt to access one of the neighbouring words within the block. Because of this locality of reference property, main memory bus traffic is reduced since the processor is likely to engage in subsequent data transactions directly with the cache. Cache accesses take less time than main memory accesses. Consequently, the use of a cache increases processor throughput.

Many modern microprocessors execute multiple instructions within the same processor clock cycle. In some instances, the processor may attempt to execute memory operations simultaneously. In those cases, the processor may require simultaneous access to multiple words stored within cache memory. Accordingly, the cache may include multiple ports, each port for conducting a separate data transaction.

A multi-port cache may be implemented as a single multi-port SRAM. However, such a configuration is very slow in operation and occupies a relatively large chip area. Alternatively, as described in U.S. Patent No. 5,359,557, issued to Aipperspach et al., a dual-port cache may be implemented with two single-port memory arrays, each corresponding to one of the cache ports. The two arrays have the same address space. This cumbersome arrangement requires complex data coherency circuitry to ensure that the arrays store the same data when data is modified at one of the cache ports. Further, the use of two arrays to store redundant copies of the same data occupies an unnecessarily large chip area.

Accordingly, there is a desire to find a smaller, more efficient means of implementing a multi-port cache memory.

The present invention provides a multi-port cache memory. The multi-port cache operates in a microprocessor system, and includes multiple memory banks and

multiple ports for enabling accesses to the banks. Conflict detection circuitry detects simultaneous addressing of a first memory bank through a first port and a second port, and stalls microprocessor operations for a predetermined number of clock cycles in response to the detection of simultaneous addressing. Conflict resolution circuitry allows access to the first bank through the first port during the stall, and allows access through the second port after the stall is complete. Generally, the conflict resolution circuitry allows access through ports that are attempting to access the first memory bank in order of ascending priority during successive clock cycles while the microprocessor is stalled. One or more of the ports attempting to access the first bank may be allowed access before or after the time the microprocessor is stalled. Each bank is single-ported. The banks have non overlapping address spaces, and are addressed so that words within a cache block are distributed among multiple banks.

The objects, features and advantages of the present invention will be apparent to one skilled in the art in light of the detailed description in which the following figures provide examples of the structure and operation of the invention:

Figure 1 illustrates a computer system having a multi-port cache of the present invention.

Figure 2 is a block diagram illustrating a processor coupled to a multi-port cache of the present invention.

Figure 3A is a timing diagram illustrating cache timing in the absence of a bank conflict.

Figure 3B is a timing diagram illustrating cache timing in the presence of a bank conflict.

The present invention provides a multi-port cache memory having multiple memory banks. In the following description, numerous details are set forth in order to enable a thorough understanding of the present invention. However, it will be understood by those of ordinary skill in the art that these specific details are not required in order to practice the invention. Further, well-known elements, devices, process steps and the like are not set forth in detail in order to avoid obscuring the present invention.

Figure 1 illustrates a computer system having a multi-port CPU 100, a main memory 102, a main memory interface 104, and a multi-port cache 106 of the present invention. The main memory interface 104 manages the information exchange between the cache 106 and main memory 102 to maintain cache coherency when a CPU access misses the cache or when the CPU writes new data into the cache. The cache 106 is shown as having

two ports, although those skilled in the art will recognize that the present invention is easily extended to a cache having any number of ports.

Preferably, the processor is capable of executing multiple parallel operations, and thus may require simultaneous access to more than one word stored within the cache. In another configuration (not shown), separate processors or other agents may each require access to a corresponding cache port.

Figure 2 is a detailed block diagram of a processor 100 coupled to an embodiment of the cache 106 of the present invention. In this example, the cache is a two-way set-associative cache. Unlike the prior art, the cache of the present invention does not employ a dual-port SRAM or redundant single-port arrays that store the same data. Instead, the present invention employs multiple single-port memory banks, where each bank stores data for a non-overlapping address space. Preferably, each bank may be accessed by any of the ports. As long as no two ports attempt to access the same bank, all ports can execute simultaneous accesses to the cache. In the event of a bank conflict, i.e., when two ports attempt to access the same bank, the cache controls the timing of the accesses as described below.

According to the present invention, the CPU 100 can issue multiple accesses to the cache 106, represented as a first address A0 and a second address A1. These addresses correspond to the two ports 201 and 203 of the cache of this example. In this example, the cache itself comprises a first bank 200, bank0, and a second bank 202, bank1. Here, each bank holds eight kilobytes (8 KB) of data, where four bytes comprise one 32-bit word. Thus, each bank stores 2K words. Further, each cache block is two words long, and two blocks comprise one set of the two-way set-associative cache of this example. Those skilled in the art will recognize that the present invention is applicable to other memory configurations, and that, in particular, the number of banks need not necessarily equal the number of ports.

Each bank is coupled to a plurality of read buses 204 through a corresponding tri-state bus driver 206, each read bus 204 corresponding to one of the ports. Each bank is further coupled to a plurality of write buses 208 through a write multiplexer 210, each write bus 208 corresponding to one of the ports.

The read and write busses 204, 208 are coupled to the input/output ports of the CPU 100 (the coupling is not shown to keep the figure simple).

The circuitry for addressing the banks is divided into address circuitry dedicated to a corresponding port and address circuitry common to both ports. In this

embodiment, each port is coupled to a dual tag RAM 212, where each tag array 214 corresponds to a way of the two-way set-associative cache. The tag from each array is fed into a corresponding comparator, 216 which compares the tag to the tag field of the corresponding port address. The resulting hit signal is passed to a corresponding port input of a hit multiplexer 218 for each bank. The hit signal here is a two-bit "one hot" signal in which at most one bit may take on a logical one value. Each bank also is coupled to a row multiplexer 220 that receives the set index field of each port address. Further, read/write control signals are passed from each CPU port to a corresponding input of a read/write multiplexer (not shown) for each bank to indicate whether a read or write memory operation is to be performed. In one embodiment, a write enable signal from each port is passed to a corresponding input of a write multiplexer. Similarly, a read enable signal from each port is passed to a corresponding input of a read multiplexer. The output of the multiplexers is coupled to write enable and read enable inputs, respectively, of the corresponding bank. The read and write multiplexers together are referred to herein as the "read/write multiplexer."

The address circuitry that is common to both ports includes conflict detect circuitry 222 that receives the bank address portion of the port addresses. In this example, each bank address passes through a 1:2 bank decoder 224, which produces a bank select signal in response. For example, if a zero bank address bit represents a selection of bank0, then the bank decoder 224 will output a one from its bank0 output and a zero from its bank1 output. The bank select signal (bd) from each port's decoder is fed into a corresponding conflict resolution circuit 226 for each bank. The output of the conflict resolution circuitry 226 controls the row multiplexer 220, the hit multiplexer 218 and the read/write multiplexer (not shown) for each bank to determine which port will have access to the bank. The conflict resolution circuitry 226 also controls the tri-state drivers 206 for the read buses 204 (Figure 2 assumes active high) and the write bus multiplexers 210 to assure access to the bus corresponding to the selected port.

In one example of the memory organization of the cache of Figure 2, each bank stores 8 KB of data with each word comprising four bytes. Each cache block comprises two words. The memory contains 1K sets with two blocks per set because the cache is a two-way set-associative cache. Bit 2 of the address selects the bank, whereas bits 3-12 select one of the sets. Bits 13-31 of the address are used in the tag comparison to indicate the presence of an addressed block in the cache.

The operation of the cache of the present invention will be described with respect to the timing diagrams of Figures 3A and 3B. Figure 3A illustrates cache timing

where there is no bank conflict. Figure 3B illustrates cache timing with a bank conflict. In both cases, the CPU attempts to perform simultaneous accesses of the cache by issuing an address A0 from a first CPU port 228 and an address A1 from a second CPU port 230. The addresses are respectively received by a first cache port 201 and a second cache port 203 over an internal CPU bus 232. In this example, during cycle 0, the second bits of the addresses are fed into the conflict detection circuitry 222 to determine whether both ports are attempting to access the same memory bank. Here, assume that A0[2] = 0 and A1[2] = 1. In that case, the bank address decoder 224 for port0 will output a bank select signal bd[0][0] = 1 to the conflict resolution circuitry 226 for bank0 200 and a bank select signal bd[0][1] = 0 to the conflict resolution circuitry 226 for bank1 202. The bank address decoder 224 for port1 201 will output a bank select signal bd[1][0] = 0 to the conflict resolution circuitry 226 for bank0 200 and a bank select signal bd[1][1] = 1 to the conflict resolution circuitry 226 for bank1 202. In cycle 0, the conflict resolution circuitry 226 determines which port input will be passed by the row multiplexer 220, the hit multiplexer 218 and the read/write multiplexer to each bank, and selects the proper read or write bus to communicate with the bank (depending upon whether a read or write operation is being performed).

For this two-port example, the conflict resolution circuitry 226 implements the following logic equations:

sel[0][i] = bd[0][i] AND NOT (sel_ctrl[1])

sel[1][i] = (NOT (bd[0][i]) AND bd[1][i]) OR sel_ctrl[1]

where the port select signal sel[j][i] gives input port j access to bank i if sel[j][i]=1. When a bank conflict occurs, the conflict resolution circuitry first allows the lower-numbered port, port0, to access the addressed bank. In that clock cycle sel_ctrl[1] =0. In the next cycle, the override signal sel_ctrl[1] takes on a value of 1 to give priority of access to port 1.

In Figure 2, the two-bit signal sel0 represents the two port-select signals for bank0, and the two-bit signal sel1 represents the two port-select signals for bank1. These combined signals select the appropriate port input to the multiplexers. Alternatively, the conflict resolution logic may be implemented by any circuitry that embodies the logic of Table 1.

PORT1		PORT0		PORT1		PORT0	
BANK1 bd[1][1]	BANK0 bd[1][0]	BANK1 bd[0][1]	BANK0 bd[0][0]	BANK1 SEL[1][1]	BANK0 SEL[1][0]	BANK1 SEL[0][1]	BANK0 SEL[0][0]
0	1	0	1	0/0	0/1	0/0	1/0
0	1	1	0	0/0	1/0	1/0	0/0
1	0	0	1	1/0	0/0	0/0	1/0
1	0	1	0	0/1	0/0	1/0	0/0

TABLE 1

In the table, "x/y" indicates that the port select signal takes on a value of x in one clock cycle followed by a value of y in a subsequent clock cycle.

In this example, bd[0][0] = 1 and bd[1][0] = 0, whereas bd[0][1] = 0 and bd[1][1] = 1. Thus, in cycle 0 of **Figure 3A**,

sel[0][0] = 1

sel[0][1] = 0

sel[1][0] = 0

sel[1][1] = 1

In the absence of a conflict, the sel_ctrl override signal is inoperative. As a result, bank0 is accessible to port0 and bank1 is accessible to port1.

In sum, the conflict resolution circuitry 226 determines which port communicates with

each bank. This selection is based upon the bank address field of the port addresses, which is bit 2 in this example. The other bits are used to address a particular word within the banks. Bits 3-12 are the set index fed into the dual tag array for each port. In this example, a set comprises two blocks, with one block in each bank. Bits 13-31 comprise the tag address field that is compared to the tags from the dual tag array 212.

If the tag comparison results in a hit in one of the arrays, the hit signal selects the word within the block. In case of a cache miss for any one of the ports, the miss is handled by loading the miss block into the cache. Operation resumes as if the miss did not occur, resulting in a hit. For example, if one instruction attempts two simultaneous accesses and one port hits while the other port misses, the miss is first handled. Then, the instruction is restarted, resulting in two hits with the conflict resolution circuitry operating as described herein.

The set index and the hit signal are routed to the correct bank through the multiplexers controlled by the conflict resolution circuitry 226. Assume hits for both port addresses. During cycle 0, the hit signal, hit0, from port0 201 is routed through the hit multiplexers 218 to the hit input of bank0 220, whereas the hit signal, hit1, from port1 203 is routed through the hit multiplexers 218 to the hit input of bank1 202. The data read from or written to port0 201 is represented by X, whereas the data read from or written to port1 203 is represented by Y. During cycle 1, both of these ports are in communication with a bank. Here, X data from port0 201 is read from or written to bank0 200, and Y data from port1 203 is read from or written to bank1 202.

Figure 3B is a timing diagram illustrating the operation of the cache of the present invention in case of a bank conflict. In this example, assume that the second bits of both port addresses equal zero, i.e., both ports attempt to access bank0. In response, the conflict detection circuitry 222 will stall the operations of the CPU 100 in the next cycle, i.e., cycle 1. The mechanism employed by the conflict detection circuitry 222 to stall the CPU can be implemented using circuitry similar to that employed by standard cache control logic to stall the CPU during a cache miss.

In this example A0[2] = A1[2] = 0. Thus, bd[0][0] = 1 and bd[1][0] =1, whereas bd[0][1] = 0 and bd[1][1] = 0. Accordingly,

sel[0][0] = 1 AND NOT (sel_ctrl1)

sel[0][1] = 0 AND NOT (sel_ctrl1)

sel[1][0] = (NOT (1) AND 1) OR sel_ctrl1

sel[1][1] = (NOT (1) AND 0) OR sel_ctrl1

The bank select signals for each bank are OR'ed together by an OR gate 250 having an output fed into a bank enable input. If no port attempts to access a bank, then the bank is not enabled. Here, bank1 is not being accessed. Consequently, the signal sel[1] (i.e., sel[0][1] and sel[1][1]) for bank1 has no effect.

However, both ports are attempting to read from bank0. Assume hits for both port addresses. During cycle 0, the hit signal, hit0, from port0 is routed through the hit multiplexers 218 to the hit_bank0 input of bank0 so that the data word X can be output from port0 during cycle 1.

Second, sel_ctrl is asserted during the stall (cycle 1) so as to force sel0 to select port1 during cycle 1. See Figure 3B and Table 1. As a result, during the stall cycle 1, the hit signal, hit1, from port1 is routed through the hit multiplexers 218 to the hit_bank0 input, of bank0 so that the data word Y can be outputted through port1 during the next cycle, cycle 2. Further, during the stall cycle, the result of a read operation for port0 is latched on the read bus for port0 by latching circuitry on the bus (not shown). As a result, data X read from port0 and data Y from port1 appear simultaneously during cycle 2. Because CPU operations are stalled during cycle 1, it appears to the CPU that the dual port cache access occurs simultaneously in a cycle immediately following cycle 0.

Note that the conflict resolution circuitry 226 grants priority access to port0 in case of a conflict. Those skilled in the art will recognize that the conflict resolution circuitry 226 may grant access to conflicting ports in any order of priority. In the examples described herein, the ports are numbered so that low-numbered ports correspond to those requiring high-priority access, whereas high-numbered ports can wait longer for access.

Further, the conflict resolution circuitry is not limited to resolving conflicts between only two ports. For a cache having K ports, if N 2 ports attempt to access the same bank, then access may first be given to the lowest numbered port, and the CPU stalled for N-1 cycles to allow access by the remaining conflicting ports in ascending order by port number. For example, for a cache with K=3 ports, for each bank i, there are three bank select signals per bank, bd[0][i], bd[1][i], bd[2][i], one for each port. There are three port select signals sel[0][i], sel[1][i], sel[2][i], indicating that port 0, 1 or 2 is selected to address the bank.

There are two selection control signals, shared by all banks, to override priorities of bank conflict resolution: sel_ctrl1, sel_ctrl2. If sel_ctrl1 is asserted, then port 1 is selected. If sel_ctrl2 is asserted, then port 2 is selected. If neither sel_ctrl1 nor sel_ctrl2 is asserted, then port 0 has priority.

For each bank i:

sel[0][i] = bd0[i] AND NOT (sel_ctrl1 OR sel_ctrl2)

sel[1][i] = (NOT (bd0[i]) AND bd1[i]) OR sel_ctrl1

sel[2][i] = (NOT (bd0[i]) AND NOT (bd1[i]) AND bd2[i]) OR sel_ctrl2

In general, for K ports, where K>3,

for each bank i:

for each port j (0 j < K):

for each bank select signal bd[j][i] of port j in bank i:

for each selection control signal sel_ctrl[m](m=1,. . ., K-1):

the conflict resolution circuitry generates output select signals sel[j][i] as follows:

sel[0][i]=bd[0][i] AND NOT (sel_ctrl[1] OR sel_ctrl[2] OR . . . OR sel_ctrl[K-1]

sel[1][i]=(NOT (bd[0][i]) AND bd[1][i]) OR sel_ctrl[1]

sel[2][i]=(NOT (bd[0][i]) AND NOT (bd[1][i]) AND bd[2][i]) OR sel_ctrl[2]

sel[j][i] = (NOT (bd[0][i])

AND NOT (bd[1][i])

. . .

AND NOT (bd[j-1][i])

AND (bd[j][i]))

OR sel_ctrl[j]

One can see that a large number of bank conflicts would give rise to many stall cycles that would hinder overall performance. Thus, it is advantageous to limit the number of bank conflicts within the same CPU cycle. According to one embodiment of the present invention, bank conflicts are avoided in the compiler and application software by allocating variables in nearby instructions to addresses in different banks. Thus, it is highly unlikely that the same bank would be addressed in the same cycle. Further, the organization of the address space itself helps to reduce the chance of a bank conflict. By using lower order address bits, e.g., the second bit, to select the bank, adjacent words of the cache block are evenly distributed among all the banks. In this manner, the addressing of adjacent words will result in the addressing of different banks. Because of the locality of reference property, this organization thus reduces the chance of conflict.

Although the invention has been described in conjunction with particular embodiments, it will be appreciated that various modifications and alterations may be made by those skilled in the art without departing from the spirit and scope of the invention. For example, the cache can be organized as an eight-way set-associative cache of eight banks. In

that configuration, address bits 6-10 act as the set index. Each set comprises two rows in each bank. Bit 5 selects one of the two rows, and bits 2-4 select the bank. The address bits 11-31 are used for the tag comparison. Bits 0-1 correspond to the byte within a word. Further, the present invention can be applied to a pipelined cache.

CLAIMS:

1. A microprocessor system with a multi-port cache comprising:

a plurality of memory banks;

a plurality of ports for enabling accesses to the banks; and

conflict detection circuitry for detecting simultaneous addressing of a first memory bank through a first port and a second port, and for stalling processor operations for a predetermined time in response to the detection of simultaneous addressing.

2. The processor system of claim 1, further comprising:

conflict resolution circuitry for allowing access to the first memory bank through the first port during the stall and for allowing access to the first memory bank through the second port after the stall is complete.

3. The processor system of claim 1, wherein each bank is single-ported.

4. The processor system of claim 1, wherein the banks are addressed so that words within a cache block are distributed among multiple banks.

5. The processor system of claim 1, wherein the banks have non overlapping address spaces.

6. A processor system according to Claim 1, 3, 4 or 5, comprising conflict resolution circuitry for allowing access to the first memory bank through ports that are attempting to access the first memory bank in order to ascending priority during successive clock cycles while the processor is stalled.

7. A multiport memory comprising

a plurality of memory banks;

a plurality of ports for enabling accesses to the banks; and

conflict detection circuitry for detecting simultaneous addressing of a first memory bank through a first port and a second port, and an output for a signal to stall processor operations for a predetermined time in response to the detection of simultaneous addressing.

8. A multiport memory according to Claim 7, conflict resolution circuitry for allowing access to the first memory bank through the first port during the stall and for allowing access to the first memory bank through the second port after the stall is complete.

9. A multiport memory according to Claim 8, comprising conflict resolution circuitry for

allowing access to the first memory bank through ports that are attempting to access the first memory bank in order of ascending priority during successive clock cycles while the processor is stalled.

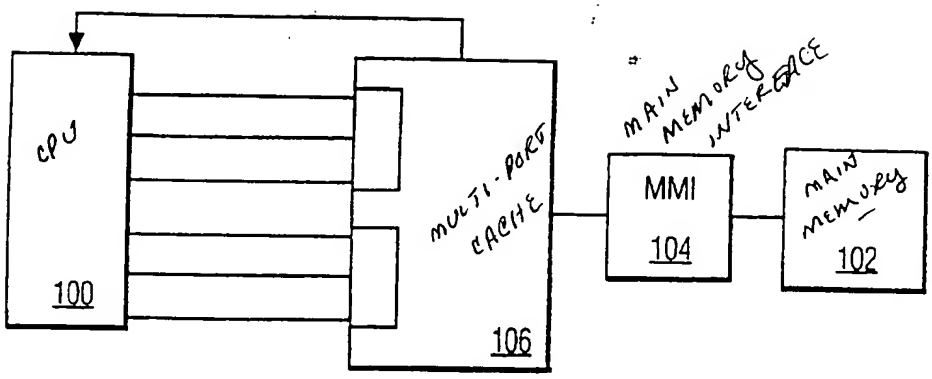

CPU
100
MULTI-PORT CACHE
106
MAIN MEMORY INTERFACE
MMI
104
MAIN MEMORY
102

FIG. 1

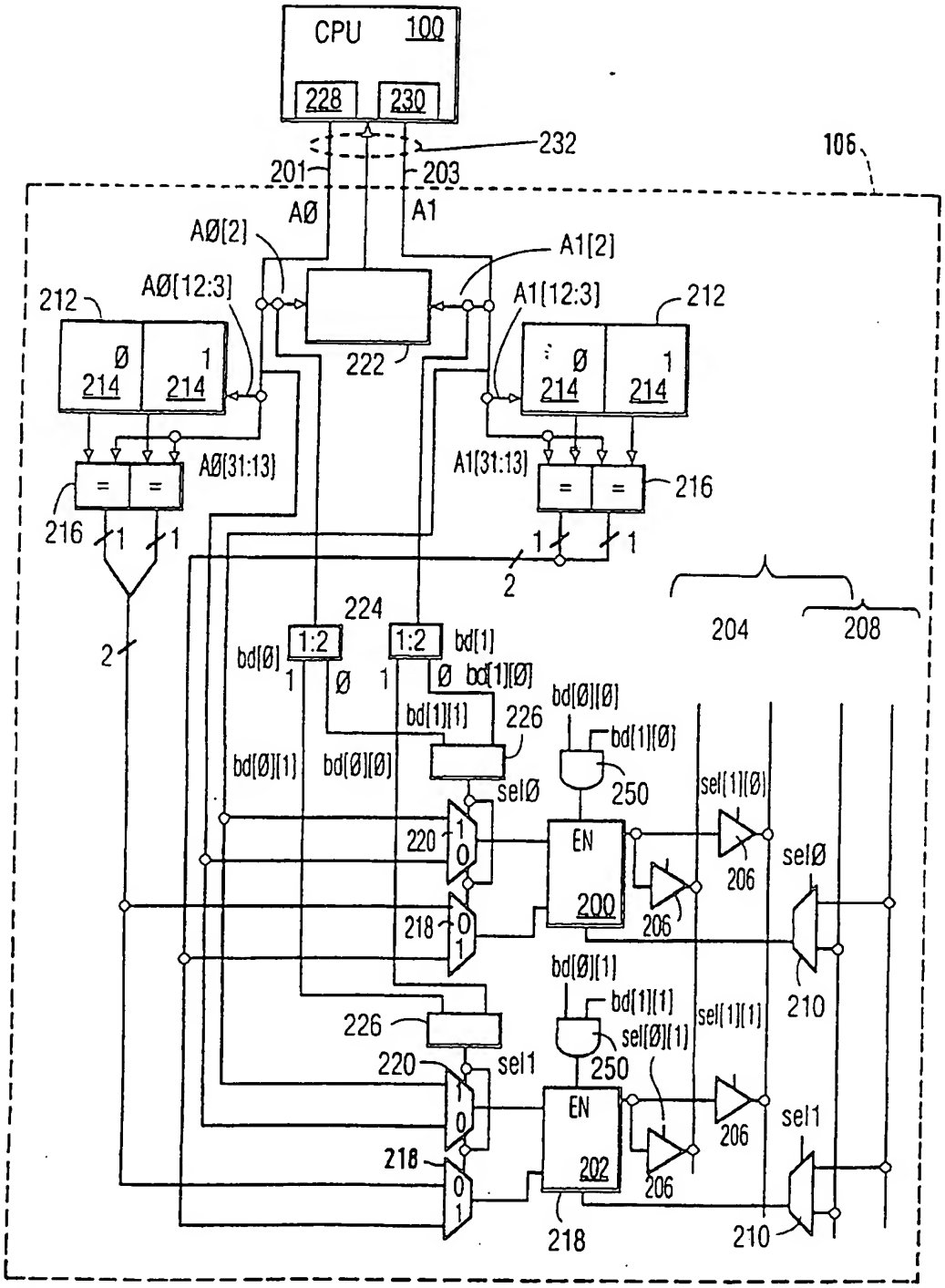

CPU 100
228
230
232
106
201
203
AØ
A1
AØ[2]
A1[2]
AØ[12:3]
A1[12:3]
212
222
Ø
1
214
AØ[31:13]
A1[31:13]
=
216
2
204
208
224
1:2
bd[Ø]
bd[1]
bd[1][Ø]
bd[1][1]
bd[Ø][1]
bd[Ø][Ø]
226
selØ
sel1
220
218
EN
200
202
250
206
sel[1][Ø]
sel[1][1]
sel[Ø][1]
210

FIG. 2

	Ø	1
AØ	AØ	
A1	A1	
STALL	Ø	Ø
sel[Ø][Ø]	1	
sel[1][1]	1	
port Ø		X
port 1		Y
sel_ctrl	Ø	Ø
HIT_BANK Ø	HIT Ø	
HIT_BANK 1	HIT 1	

FIG. 3A

	Ø	1	2
AØ	AØ		
A1	A1		
STALL	Ø	1	Ø
sel[Ø][Ø]	1	Ø	
sel[1][1]	Ø	1	
port Ø		X	X
port 1			Y
sel_ctrl	Ø	1	Ø
HIT_BANK Ø	HIT Ø	HIT 1	
HIT_BANK 1			

FIG. 3B